

Programming Assignment #2

COEN 283 Operating Systems
Department of Computer Science and Engineering
Santa Clara University

Dr. Ming-Hwa Wang
Phone: (408) 805-4175
Course website: <http://www.cse.scu.edu/~m1wang/os/>
Office Hours: Tuesday & Thursday 9:00-9:30pm, Saturday 11:00am-12:00pm

Fall Quarter 2024

Email address: m1wang@scu.edu

<http://www.cse.scu.edu/~m1wang/os/>

Due date: 11:59pm October 27, 2024

Banker's Algorithm (200 points)

Please implement the banker's algorithm to handle multiple resources using Java (not preferable), C/C++, or Go. The input consists of multiple lines, 1) the first line is total existing resources for each type. 2) Followed by lines for each process and contains process id, how many instances of each resource are currently assigned to the process, and how many more instances of each resource are still needed in order to complete the process. 3) Then followed by requests, one for each line. The second and the third can be in any order. If the input does not follow the 3 formats, report error and exit, and if the input is the third case and either the pid is not exist or requested more than needed, report warning, ignore it and continue for the next input if any. The sums of all assigned resource should not exceed the maximum resources, otherwise, report error and exit. For example,

```
# existing maximum resources
6 3 4 2
# pid assigned needed
13512 assigned 3 0 1 1 need 1 1 0 0
5646 assigned 0 1 0 0 need 0 1 1 2
13665 assigned 1 1 0 1 need 0 0 1 0
3206 assigned 0 0 0 0 need 2 1 1 0
# request
5646 request 0 0 1 0
# pid assigned needed
25553 assigned 1 1 1 0 need 3 1 0 0
# request
3206 request 0 0 1 0
```

Assume that the order of the number of instances for each resource type is consistent in the input, however, the number of resource types is not fixed for different runs. Any input after the sharp sign ('#') till the end of line will be comments. keywords are case sensitive (assigned, need, request), and numbers

must be separated by white spaces (spaces or tab), and leading/trailing white space can be ignored. That is, you need to support whitespace convention.

The output should first assert if the current state (before the request after the assigned/need input) is safe. Note that, deciding if the current state is safe or not depends on the order of fulfill a need, i.e., you need try all possible orders. Then report if the first request is safe (and granted if safe, denied if unsafe), and then the second request, and so on. One possible output looks like:

```
current state is safe
5646 request is granted
current state is safe
3206 request is denied
```

Student Name:

SSN/ID:

Score:

Correctness and boundary condition (60%):

Whitespace and free format compliance (5%):

Error Handling (5%):

Compiling without warning/error (5%):

Modular design, file/directory organizing, showing input, documentation, coding standards, sympathy/typing point with README (20%):

Automation (5%):

Subtotal:

Late penalty (20% per day):

Special service penalty (5%):

Total score: